418383: Game Programming
XNA Content Pipeline

Pramook Khungurn

pramook@gmail.com

Content

- Content = เนื้อหา
  - รูป
  - เพลง
  - เสียง
  - 3D Model
  - แผนที่
  - ฯลฯ
- ว่าไปคือสิ่งที่ศิลปินสร้าง
- อ่านเข้ามาในเกมในเมธอด LoadContent

เป้าหมายของวันนี้

- เขียนโปรแกรมให้
  - เราสามารถประกาศคลาสสำหรับแทน content ของตนเอง
  - เราสามารถใส่ content นั้นในโฟลเดอร์ Content ของเกมได้
  - เราสามารถอ่าน content นั้นด้วย ContentManager.Load ด้วย

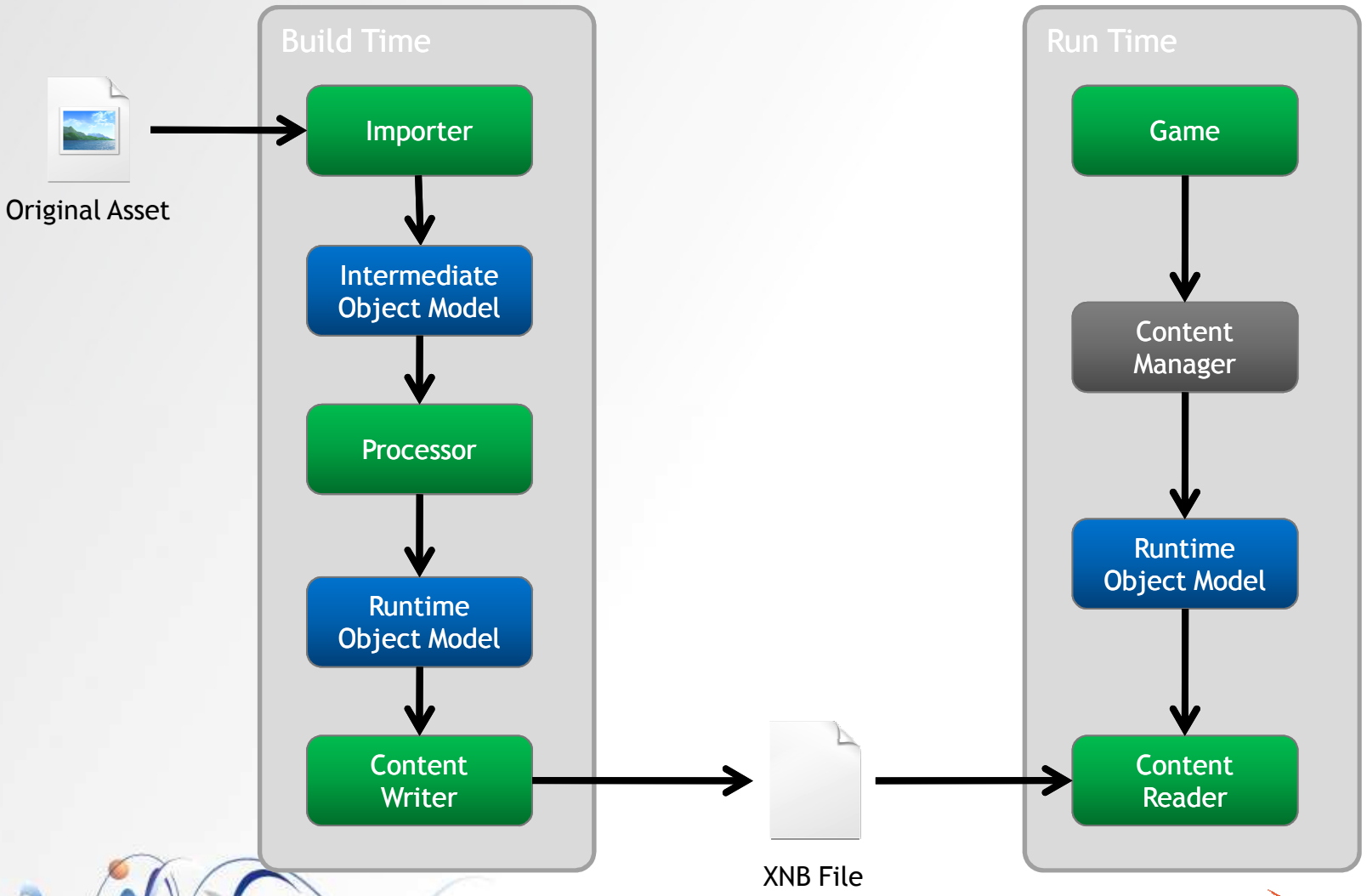
Content ที่เราเคยใช้

- Texture2D
  - `whiteTexture = Game.Content.Load<Texture2D>("white");`
- SpriteFont
  - `font = Game.Content.Load<SpriteFont>("Vera32");`
- โหลดผ่านเมธอด Load ของคลาส ContentManager

XNA Content Pipeline

- ขั้นตอนอ่าน content เข้าสู่เกม
- การจัดการ content ใน XNA แบ่งเป็น 4 ขั้นตอน
  - การ Import
    - อ่านไฟล์ที่ศิลปินสร้าง (.jpg, .png, .3ds, ฯลฯ)
  - การ Process
    - ประมวลผล content ที่อ่านมา
    - ส่วนมากใช้สำหรับแปลง content เป็นคลาสที่ใช้จริง
  - การ Write
    - เขียน content ที่เป็นผลลัพธ์ของ processor ลงไฟล์ .xnb
  - การ Read
    - อ่านไฟล์ .xnb ก่อนนำมาใช้จริง
    - ถูกเรียกใช้ตอนสั่ง Content.Load

Content Pipeline Flow



ตัวอย่าง content

- รายละเอียดของตัวละครเกม RPG

```
public class CharacterInfo
{
    private string name;
    private int hitPoint;
    private int strength;
    private int agility;
    private int defense;
    private int magic;
      :
      :
}
```

ข้อมูลที่ศิลปินสร้างมาให้

- ข้อมูลประเภท XML (merlin.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<character>
   <name>Merlin</name>
   <strength>33</strenght>
   <agility>24</agility>
   <defense>46</defense>
   <magic>51</magic>
</character>
```

Content Pipeline Extension Library (CPEL)

- ไลบรารีสำหรับประมวลผลและเขียน content สำหรับเนื้อหาที่อยู่ในรูปแบบที่เรากำหนดได้เอง
- ใช้งานตอนคอมไพล์โปรแกรม
- เป็น project แบบหนึ่งที่ XNA อนุญาตให้เราสร้างได้

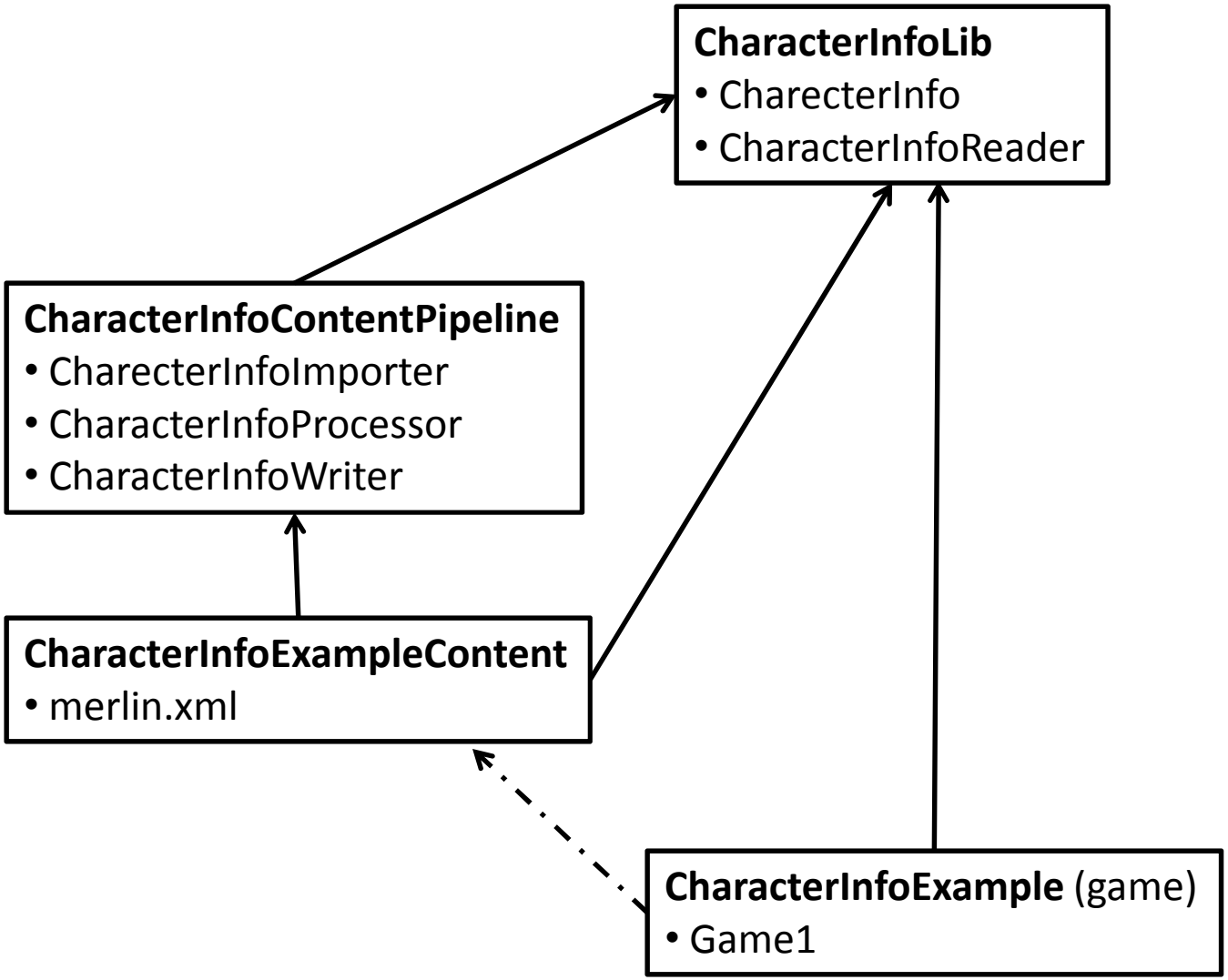
ขั้นตอนการสร้างระบบ content ของเราเอง

- สร้างเกมไลบรารีที่มีคลาสของ content นั้น (A)
  - ไลบรารีนี้ควรมีคลาสสำหรับอ่าน content นั้นด้วย
- สร้าง CPEL (B)
  - มี reference ไปยัง A
  - Importer
  - Processor
  - Writer

ขั้นตอนการสร้างระบบ content ของเราเอง

- สร้าง content project (C)
  - มี reference ไปยังสองไลบรารีแรก
- สร้าง game
  - มี reference ไปยัง A
  - **ห้ามมี reference โดยตรงไปยัง CPEL**
  - มี content reference ไปยัง C

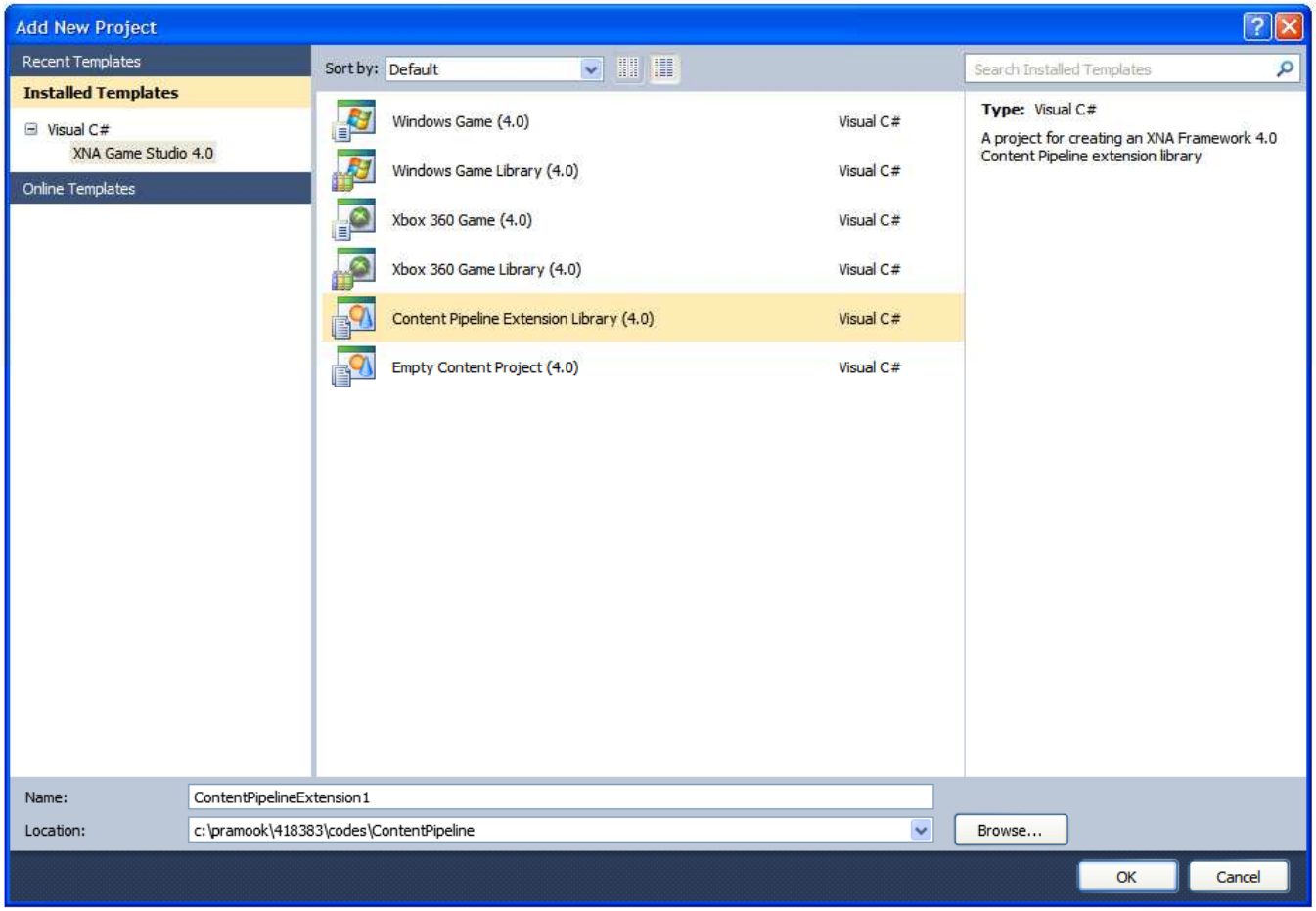
References ระหว่างส่วนประกอบของโปรแกรม



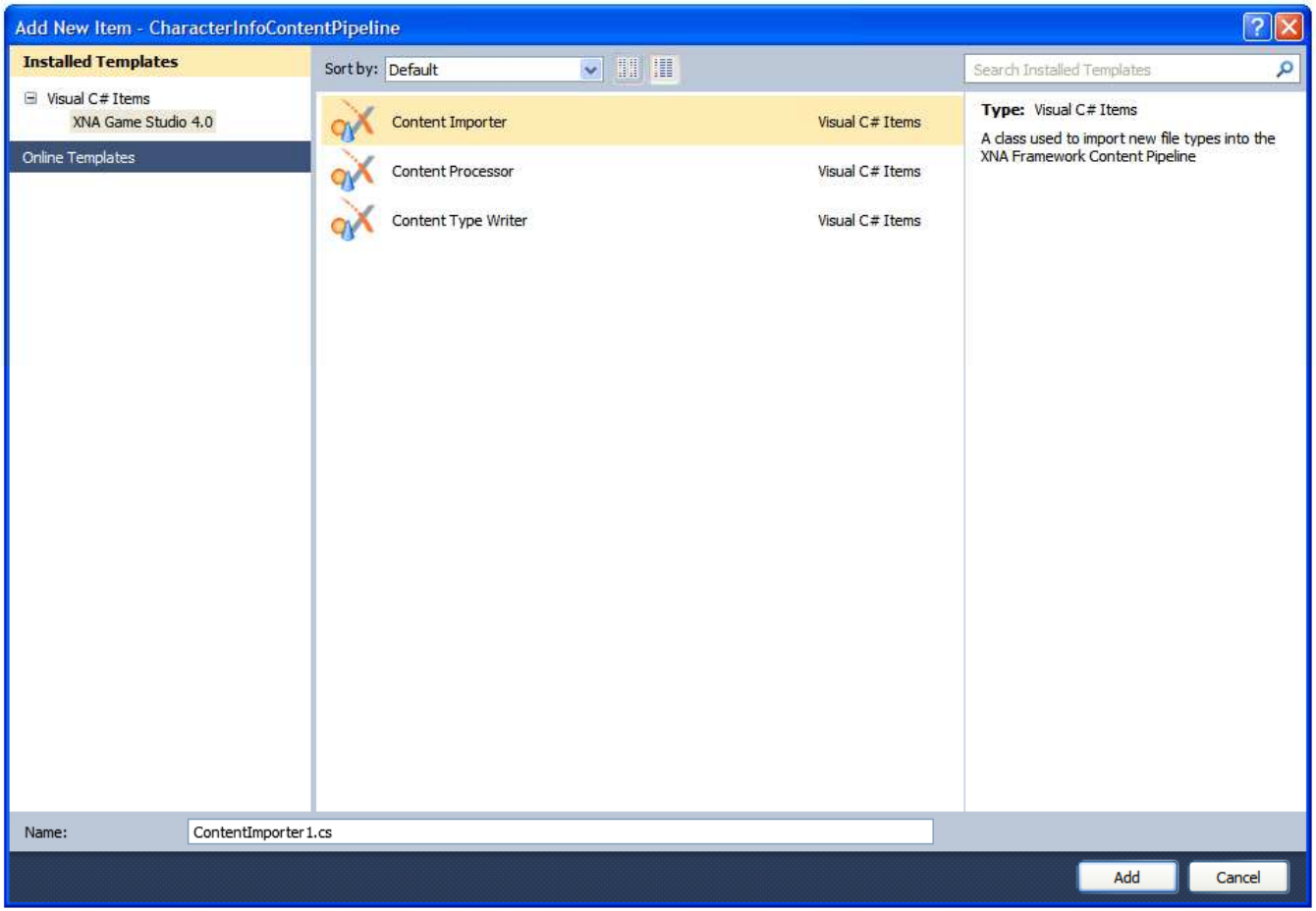
ทำไมถึงต้องไม่มี reference จากเกมไปยัง CPEL?

- เวลาแจกจ่ายเกม เราส่ง XNA Redistribuble ให้กับผู้ใช้
- แต่โค้ดที่ใช้พัฒนา CPEL ไม่ได้อยู่ใน XNA Redistributable
  - มันอยู่ใน XNA Game Studio
- ดังนั้นถ้ามี reference จากเกมไป CPEL
  เกมจะรันบนเครื่องผู้ใช้ไม่ได้

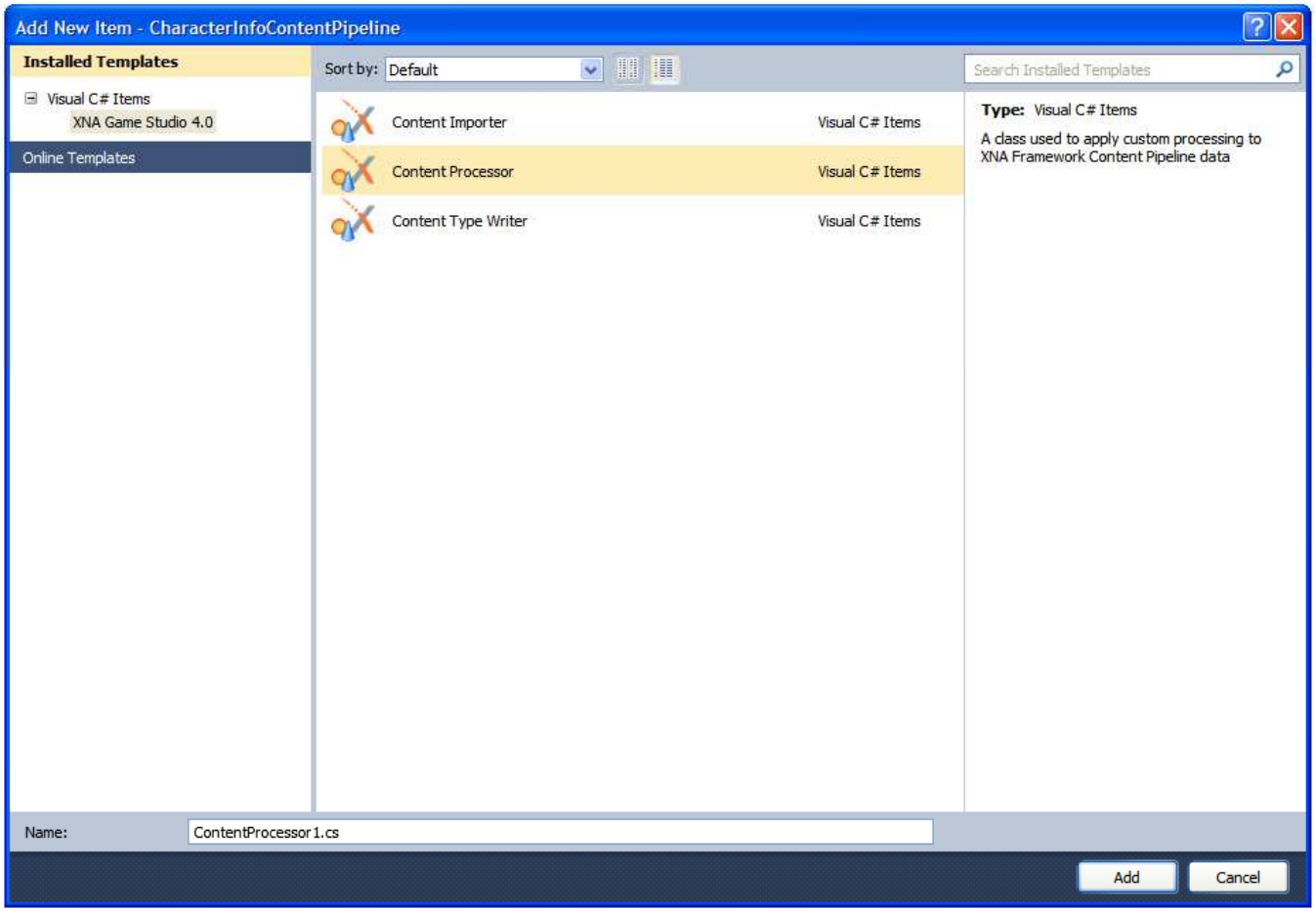
สร้าง Content Pipeline Extension Library



สร้าง Importer



สร้าง Processor



โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

โค้ด Importer ที่ XNA สร้างให้

ชนิดของข้อมูลที่

content importer

อ่านจากไฟล์

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

นามสกุลของไฟล์ content

โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

นามสกุลของไฟล์ content

โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

ชื่อของ content importer

โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

Processor ที่จะรับ content ไปประมวลผลต่อ

โค้ด Importer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to import.
using TImport = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".abc",
       DisplayName = "ABC Importer",
       DefaultProcessor = "AbcProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename, ContentImporterContext context)
        {
            // TODO: read the specified file into an instance of the imported type.
            throw new NotImplementedException();
        }
    }
}
```

เมธอดสำหรับอ่าน file

โค้ด Processor ที่ XNA สร้างให้

```
// TODO: replace these with the processor input and output types.
using TInput = System.String;
using TOutput = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentProcessor(DisplayName =
    "CharacterInfoContentPipeline.CharacterInfoProcessor")]
    public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
    {
        public override TOutput Process(TInput input, ContentProcessorContext context)
        {
            // TODO: process the input object, and return the modified data.
            throw new NotImplementedException();
        }
    }
}
```

โค้ด Processor ที่ XNA สร้างให้

```
// TODO: replace these with the processor input and output types.
using TInput = System.String;
using TOutput = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentProcessor(DisplayName =
    "CharacterInfoContentPipeline.CharacterInfoProcessor")]
    public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
    {
        public override TOutput Process(TInput input, ContentProcessorContext context)
        {
            // TODO: process the input object, and return the modified data.
            throw new NotImplementedException();
        }
    }
}
```

ชนิดข้อมูลที่ importer ส่งมา

โค้ด Processor ที่ XNA สร้างให้

```
// TODO: replace these with the processor input and output types.
using TInput = System.String;
using TOutput = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentProcessor(DisplayName =
    "CharacterInfoContentPipeline.CharacterInfoProcessor")]
    public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
    {
        public override TOutput Process(TInput input, ContentProcessorContext context)
        {
            // TODO: process the input object, and return the modified data.
            throw new NotImplementedException();
        }
    }
}
```

ชนิดข้อมูลที่ processor ส่งออก

โค้ด Processor ที่ XNA สร้างให้

```
// TODO: replace these with the processor input and output types.
using TInput = System.String;
using TOutput = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentProcessor(DisplayName =
    "CharacterInfoContentPipeline.CharacterInfoProcessor")]
    public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
    {
        public override TOutput Process(TInput input, ContentProcessorContext context)
        {
            // TODO: process the input object, and return the modified data.
            throw new NotImplementedException();
        }
    }
}
```

ชื่อ content processor

โค้ด Processor ที่ XNA สร้างให้

```
// TODO: replace these with the processor input and output types.
using TInput = System.String;
using TOutput = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentProcessor(DisplayName =
    "CharacterInfoContentPipeline.CharacterInfoProcessor")]
    public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
    {
        public override TOutput Process(TInput input, ContentProcessorContext context)
        {
            // TODO: process the input object, and return the modified data.
            throw new NotImplementedException();
        }
    }
}
```

เมธอดสำหรับประมวลผล content

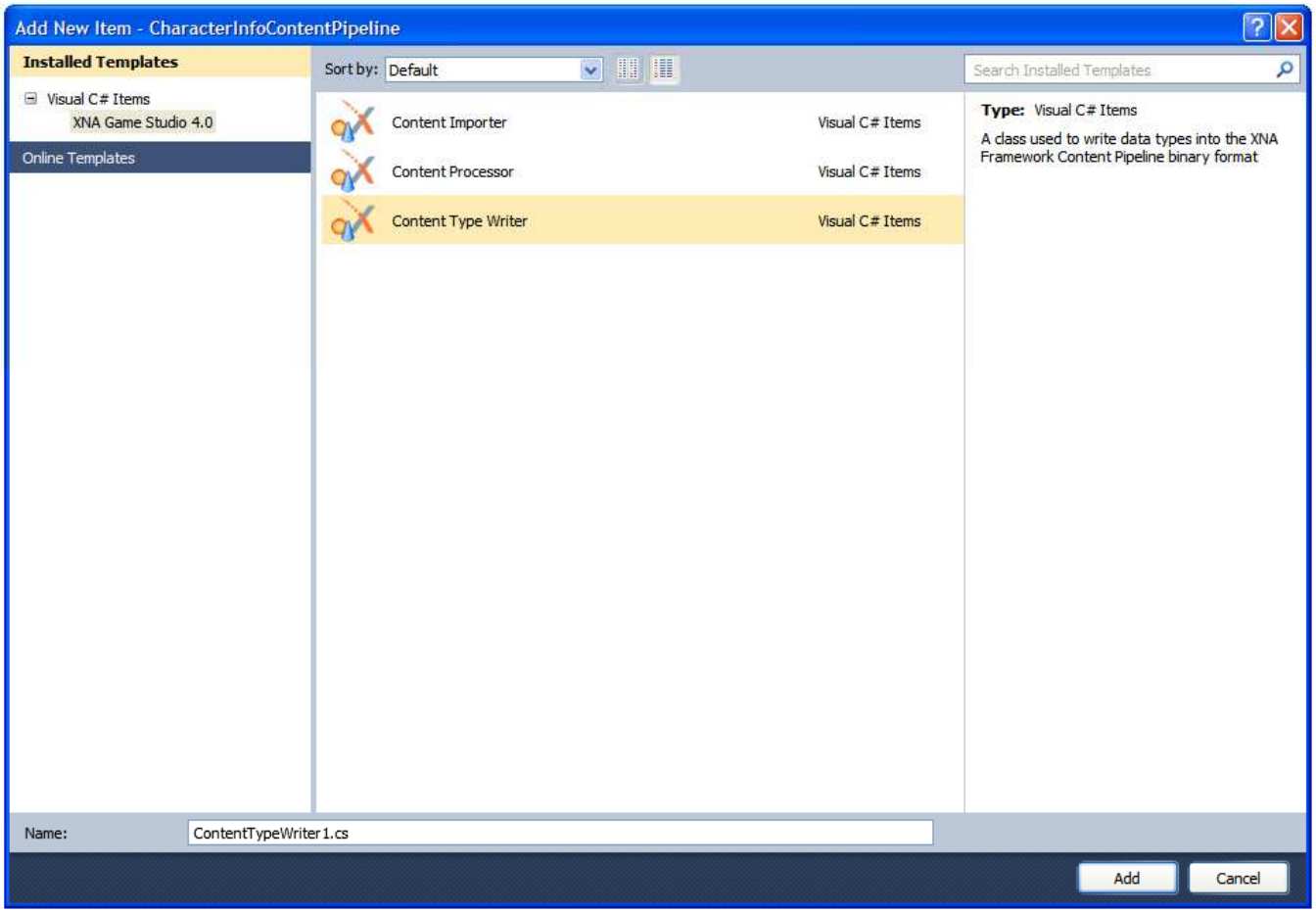
แก้ไข Importer ให้อ่านเอกสาร XML

```
using System.Xml;

using TImport = System.Xml.XmlDocument;

namespace CharacterInfoContentPipeline
{
    [ContentImporter(".xml",
        DisplayName = "Character Info Importer",
        DefaultProcessor = "CharacterInfoProcessor")]
    public class CharacterInfoImporter : ContentImporter<TImport>
    {
        public override TImport Import(string filename,
            ContentImporterContext context)
        {
            XmlDocument doc = new XmlDocument();
            doc.Load(filename);
            return doc;
        }
    }
}
```

แก้ Processor ให้แปลงเอกสาร XML เป็น CharacterInfo

```
using TInput = System.Xml.XmlDocument;
using TOutput = CharacterInfoLib.CharacterInfo;

namespace CharacterInfoContentPipeline
{
   [ContentProcessor(DisplayName = "CharacterInfoContentPipeline.CharacterInfoProcessor")]
   public class CharacterInfoProcessor : ContentProcessor<TInput, TOutput>
   {
      public override TOutput Process(TInput input, ContentProcessorContext context)
      {
         CharacterInfo charInfo = new CharacterInfo();

         XmlElement characterElement = (XmlElement)input.DocumentElement;

         charInfo.Name = characterElement["name"].FirstChild.Value;
         charInfo.Strength = Convert.ToInt32(characterElement["strength"].FirstChild.Value);
         charInfo.Agility = Convert.ToInt32(characterElement["agility"].FirstChild.Value);
         charInfo.Defense = Convert.ToInt32(characterElement["defense"].FirstChild.Value);
         charInfo.Magic = Convert.ToInt32(characterElement["magic"].FirstChild.Value);

         return charInfo;
      }
   }
}
```

สร้าง Writer


Add New Item - CharacterInfoContentPipeline
Installed Templates
Visual C# Items
XNA Game Studio 4.0
Online Templates
Sort by: Default
Search Installed Templates
Content Importer
Visual C# Items
Content Processor
Visual C# Items
Content Type Writer
Visual C# Items
Type: Visual C# Items
A class used to write data types into the XNA Framework Content Pipeline binary format
Name:
ContentTypeWriter1.cs
Add
Cancel

โค้ด Writer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to write out.
using TWrite = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            // TODO: write the specified value to the output ContentWriter.
            throw new NotImplementedException();
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            // TODO: change this to the name of your ContentTypeReader
            // class which will be used to load this data.
            return "MyNamespace.MyContentReader, MyGameAssembly";
        }
    }
}
```

โค้ด Writer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to write out.
using TWrite = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            // TODO: write the specified value to the output ContentWriter.
            throw new NotImplementedException();
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            // TODO: change this to the name of your ContentTypeReader
            // class which will be used to load this data.
            return "MyNamespace.MyContentReader, MyGameAssembly";
        }
    }
}
```

ชนิดข้อมูลที่จะเขียน

โค้ด Writer ที่ XNA สร้างให้

เมธอดสำหรับเขียนข้อมูล

```
// TODO: replace this with the type you want to write out.
using TWrite = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            // TODO: write the specified value to the output ContentWriter.
            throw new NotImplementedException();
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            // TODO: change this to the name of your ContentTypeReader
            // class which will be used to load this data.
            return "MyNamespace.MyContentReader, MyGameAssembly";
        }
    }
}
```

โค้ด Writer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to write out.
using TWrite = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            // TODO: write the specified value to the output ContentWriter.
            throw new NotImplementedException();
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            // TODO: change this to the name of your ContentTypeReader
            // class which will be used to load this data.
            return "MyNamespace.MyContentReader, MyGameAssembly";
        }
    }
}
```

เมธอดสำหรับคืนชื่อของ Reader

โค้ด Writer ที่ XNA สร้างให้

```
// TODO: replace this with the type you want to write out.
using TWrite = System.String;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            // TODO: write the specified value to the output ContentWriter.
            throw new NotImplementedException();
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            // TODO: change this to the name of your ContentTypeReader
            // class which will be used to load this data.
            return "MyNamespace.MyContentReader, MyGameAssembly";
        }
    }
}
```

ชื่อและ asssemby ของ Reader

แก้ Writer ให้เขียนข้อมูลลงไฟล์ .xnb

```
using TWrite = SampleLib.CharacterInfo;

namespace CharacterInfoContentPipeline
{
    [ContentTypeWriter]
    public class CharacterInfoWriter : ContentTypeWriter<TWrite>
    {
        protected override void Write(ContentWriter output, TWrite value)
        {
            output.Write(value.Name);
            output.Write(value.Strength);
            output.Write(value.Agility);
            output.Write(value.Defense);
            output.Write(value.Magic);
        }

        public override string GetRuntimeReader(TargetPlatform targetPlatform)
        {
            return "CharacterInfoContentPipeline.CharacterInfoReader,
                    CharacterInfoContentPipeline";
        }
    }
}
```

สร้าง Reader

- ต้องสร้างเอง
- คลิกขวาที่ชื่อ Project แล้ว Add New Item… → Class…
- ควรอยู่ในไลบรารีเดียวกับคลาสที่เราต้องการอ่าน

- คลาสที่สร้างต้องเป็น subclass ของ ContentTypeReader< คลาสที่คุณต้องการอ่าน > ซึ่งอยู่ใน Microsoft.XNA.Framework.Content

- ต้องมี method

  คลาสของคุณ Read(
  ContentReader input,
  คลาสของคุณ existingInstance);

โครง Reader

```
namespace CharacterInfoContentPipeline
{
    class CharacterInfoReader : ContentTypeReader<SampleLib.CharacterInfo>
    {
        protected override SampleLib.CharacterInfo Read(
            ContentReader input,
            SampleLib.CharacterInfo existingInstance)
        {
            return null;
        }
    }
}
```

แก้ Reader ให้อ่านข้อมูลจากไฟล์ .xnb

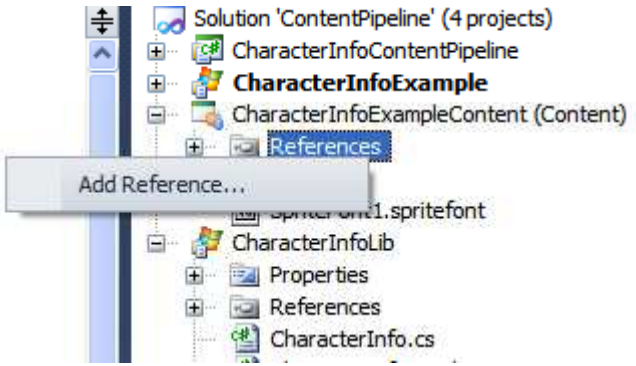
```
namespace CharacterInfoContentPipeline
{
    class CharacterInfoReader : ContentTypeReader<SampleLib.CharacterInfo>
    {
        protected override SampleLib.CharacterInfo Read(
            ContentReader input,
            SampleLib.CharacterInfo existingInstance)
        {
            CharacterInfo charInfo = new CharacterInfo();

            charInfo.Name = input.ReadString();
            charInfo.Strength = input.ReadInt32();
            charInfo.Agility = input.ReadInt32();
            charInfo.Defense = input.ReadInt32();
            charInfo.Magic = input.ReadInt32();

            return charInfo;
        }
    }
}
```

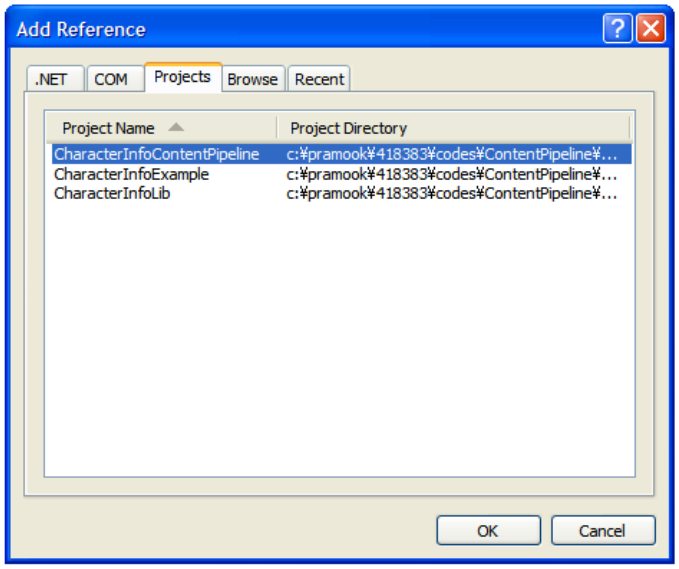
ContentWriter และ ContentReader

- ใช้สำหรับเขียนข้อมูลและอ่านข้อมูลจากไฟล์ .xnb
- ContentWriter มีเมธอด Write ใช้เขียนข้อมูลพื้นฐาน (string, int, long, float, double)
- ContentReader มีเมธอด ReadXXX สำหรับอ่านข้อมูลพื้นฐานที่ ContentWriter เขียน
  - ReadString, ReadInt32, ReadFloat, ฯลฯ
- เวลาเขียน Content Pipeline ต้องทำให้ลำดับการอ่านตรงกับลำดับที่เขียน

นำไปใช้

- เพิ่ม Content Pipeline ที่เราสร้างเข้าใน Reference ของ Content ในเกมที่จะสร้าง

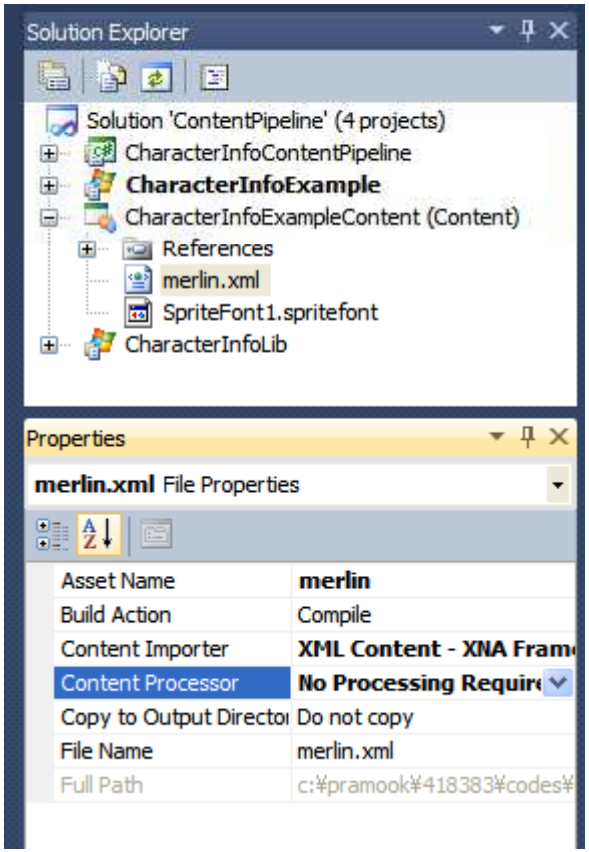




- นอกจากนี้ให้เพิ่มมันลงใน Reference ของตัวเกมเองด้วย

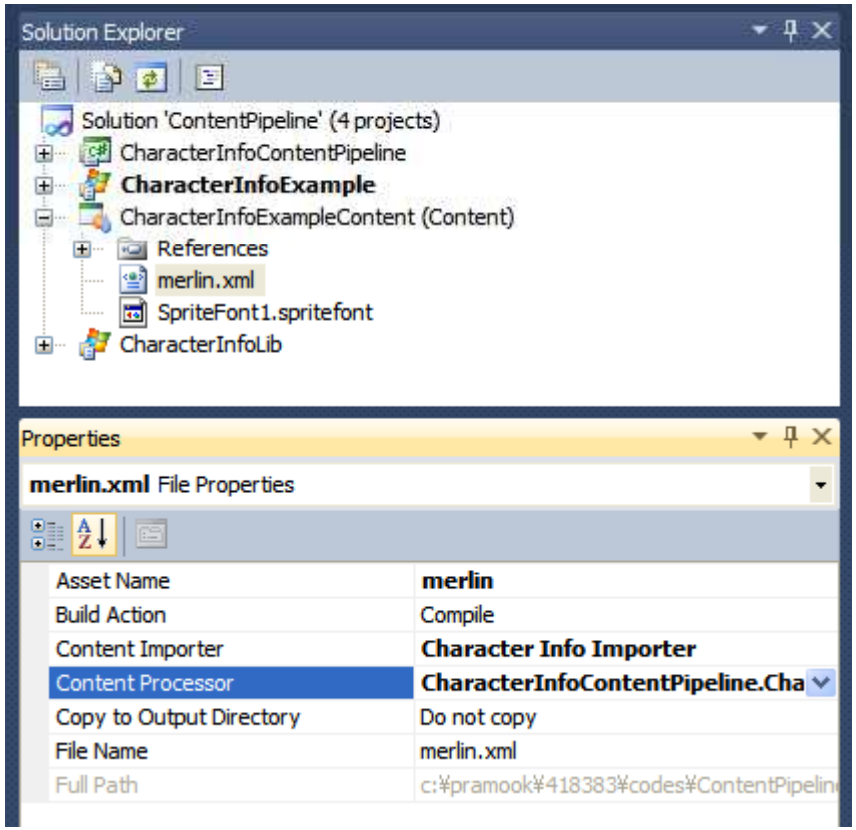
นำไปใช้

- หลังจากนั้นให้ import file content ที่ศิลปินสร้างให้



นำไปใช้

- เซต Content Import และ Content Processor

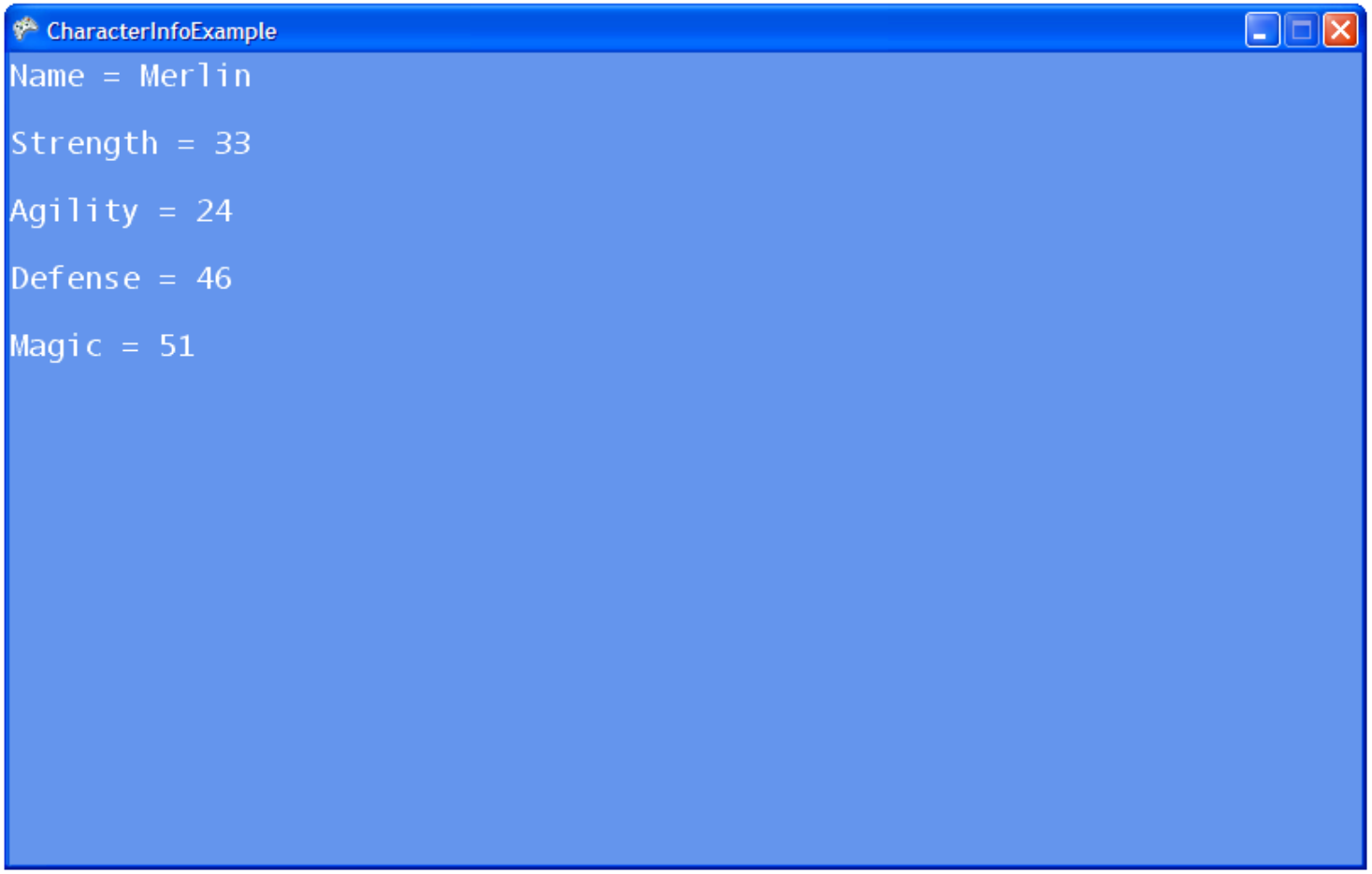


นำไปใช้

- โหลด content ด้วย Content.Load ในเมธอด LoadContent

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
    merlin = Content.Load<CharacterInfo>("merlin");
}
```

นำไปใช้


CharacterInfoExample
Name = Merlin
Strength = 33
Agility = 24
Defense = 46
Magic = 51

XML ใน C#

XML

- Extensible Markup Language
- ใช้สำหรับเก็บข้อมูลที่มีโครงสร้างเป็นต้นไม้
- ประกอบด้วย
  - tag
  - ข้อมูลซึ่งแทรกอยู่ระหว่าง tag

Tag

- ขึ้นต้นด้วย “<“ และจบด้วย “>”

- มีสามแบบ
  - Start tag เช่น <selection>
  - End tag เช่น </selection>
  - Empty-element tag เช่น <line-break/>

- Start tag กับ end tag จะอยู่คู่กัน โดย start จะมาก่อนเสมอ

Tag

- ระหว่าง start tag กับ end tag อาจมี
  - ข้อความ
  - tag อื่นๆ

- ด้วยเหตุนี้ XML ถึงสามารถใช้แทนข้อมูลที่โครงสร้างเป็นต้นไม้ได้

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

Document Type Declaration (DTD)

- ข้อมูลเกี่ยวกับรูปแบบของตัวเอกสารเอง
- ส่วนมากจะบอกว่า
  - ใช้ XML version อะไร
  - encoding ที่ใช้ในเอกสารคืออะไร
- เริ่มต้นด้วย <?xml version=
- จบด้วย ?>

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

DTD

Element

- ส่วนประกอบของเอกสารที่
  - เริ่มต้นจาก start tag แล้วจบด้วย end tag หรือ
  - มี empty element tag แค่อันเดียว

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

element

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

element

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

element

ตัวอย่าง

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>-<date>1512</date>.
  </caption>
</painting>
```

element

Attribute

- คู่ของชื่อกับค่าใน start tag และ empty-element tag

- ตัวอย่าง

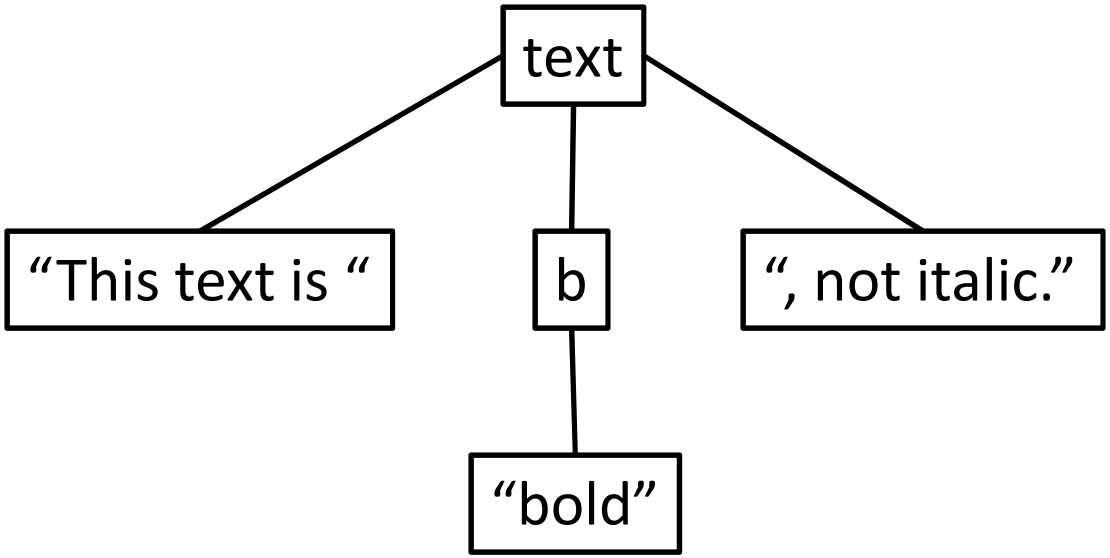
```
<img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
```

มี 2 attributes

  - src
  - alt

มอง XML เป็นต้นไม้

```
<text>This text is <b>bold</b>, not italic.</text>
```



XML ใน C#

- มี namespace ชื่อ System.Xml
- คลาสที่ใช้บ่อย
  - XmlDocument
  - XmlElement
  - XmlNode

XmlDocument

- แทนเอกสาร XML ทั้งเอกสาร

- public virtual void LoadXml(string sml);
  - เอาเอกสาร XML ที่อยู่ใน string ที่ให้มาเป็นเนื้อหาของ XmlDocument ตัวนี้
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<hello>abc</hello>");
```

XmlDocument

- public virtual void Save(string filename);
  - เซฟ XmlDocument ลงในไฟล์ที่มีชื่อตามที่กำหนดให้
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml(“<hello>abc</hello>”);
doc.Save(“sample.xml”);
```

XmlDocument

- public virtual void Load(string filename);
  - อ่านไฟล์ที่มีชื่อตามที่กำหนดให้ แล้วเอาเนื้อหามาใส่ใน XmlDocument
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.Load(“sample.xml”);
```

XmlDocument

- public XmlDocument DocumentElement;
  - คืน root element ของ XmlDocument
  - Root element คือ element ที่ element อื่นอยู่ข้างในมันทั้งหมด (ไม่รวม DTD)
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<hello>abc</hello>");
XmlDocument root = doc.DocumentElement;
```

XmlElement

- public string Name;
  - ชื่อของ tag ของ element นั้น
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml(“<hello>abc</hello>”);
XmlDocument root = doc.DocumentElement;
string name = root.Name; // ได้ “hello”
```

XmlNode

- คลาสบรรพบุรุษของส่วนประกอบของเอกสาร XML ต่างๆ
  - ข้อความ
  - XmlElement
  - XmlDocument
  - XmlAttribute
  - ฯลฯ

XmlNode

- public XmlNodeType NodeType;
  - คืนชนิดของ node มาให้
  - ค่าที่คืนมามีชนิดเป็น enum XmlNodeType
  - ค่าที่เป็นไปได้ เช่น XmlNodeType.Text, XmlNodeType.Element, XmlNodeType.Document

XmlNode

- public XmlNode FirstChild;
  - ลูกตัวแรกของ node นั้น
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<hello>abc</hello>");
XmlDocument root = doc.DocumentElement;
XmlNode firstChild = root.FirstChild;
```

ได้เป็น node แบบ text ที่เก็บข้อความ "abc" อยู่

XmlNode

- public string Value;
  - ข้อความที่เป็นค่าของ node
  - ใช้ได้กับ node ประเภท text
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<hello>abc</hello>");
XmlDocument root = doc.DocumentElement;
string text = root.FirstChild.Value; // ได้ "abc"
```

XmlNode

- public XmlNode this[string name];
  - เอา element ลูกตัวแรกที่มีชื่อของ tag เท่ากับ name
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml(“<a><b>1</b><c>2</c><b>3</b></a>”);
XmlDocument root = doc.DocumentElement;
XmlNode firstB = root[“b”];
```

ได้ node ที่แทน 1

XmlNode

- public XmlNodeList getElementsByTagName(string name);
  - คืนลิสต์ของ element ลูกทั้งหมดที่มีชื่อ text เท่ากับ name
  - เราสามารถเรียกดูลูกตัวที่ i ได้โดยการเรียกเมธอด Item(i)
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<a><b>1</b><c>2</c><b>3</b></a>");
XmlDocument root = doc.DocumentElement;
XmlNodeList bList = root.getElementsByTagName("b");
XmlNode b1 = bList.Item(0); // <b>1</b>
XmlNode b2 = bList.Item(1); // <b>3</b>
```

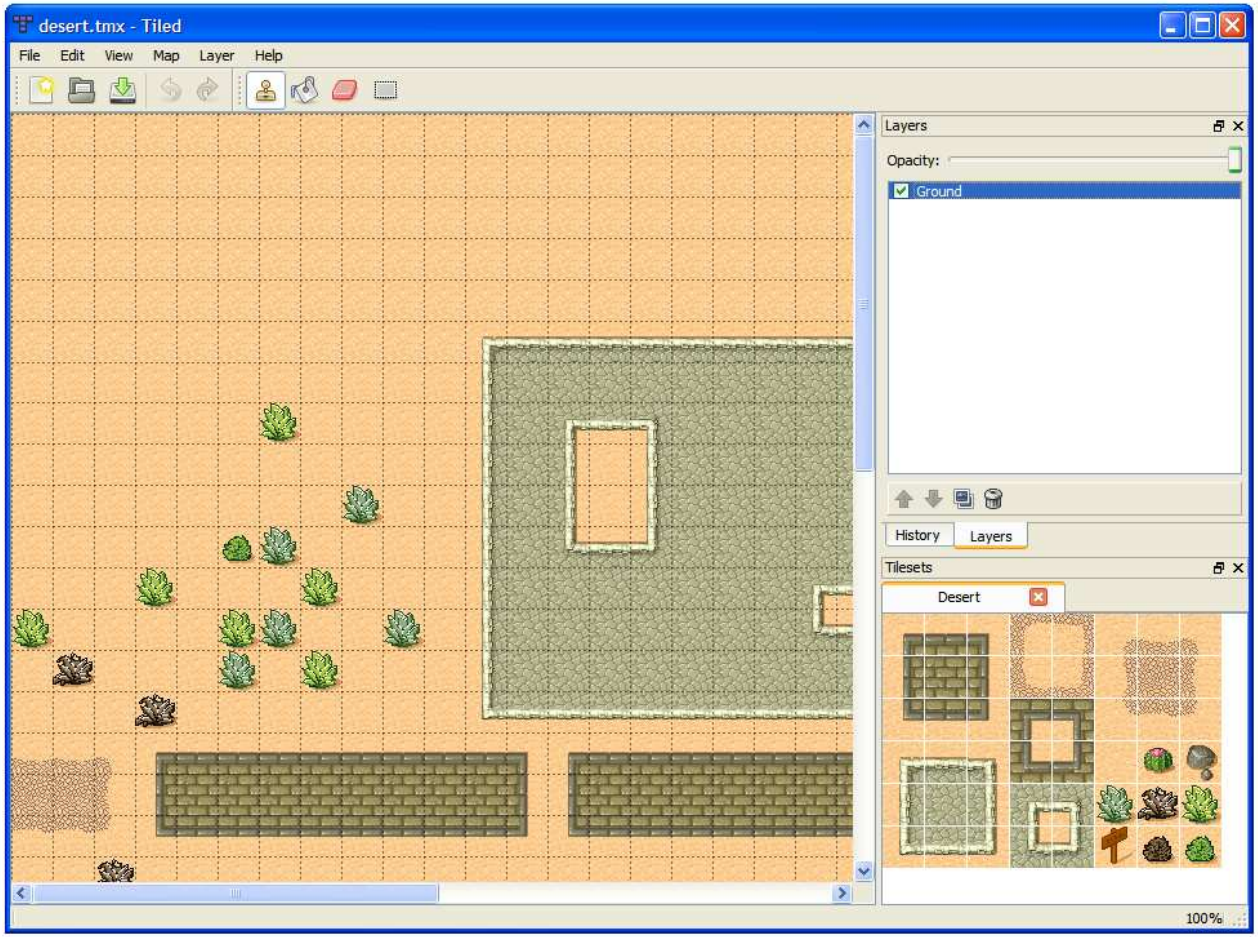
XmlElement

- public string GetAttribute(string name);
  - คืนข้อความที่เป็นค่าของ attribute ที่มีชื่อตามที่กำหนด
  - ตัวอย่าง

```
XmlDocument doc = new XmlDocument();
doc.LoadXml("<hello a="x" b="y" />");
XmlDocument root = doc.DocumentElement;
string aval = root.GetAttribute("a");
string bval = root.GetAttribute("b");
```

แผนที่สองมิติ

Tiled


desert.tmx - Tiled
File
Edit
View
Map
Layer
Help
Layers
Opacity:
Ground
History
Layers
Tilesets
Desert
100%

Tiled

- โปรแกรมสำหรับสร้างแผนที่สองมิติโดยเฉพาะ
- ข้อดี
  - แบ่งแผนที่ออกได้เป็นหลาย layer
  - สามารถสร้าง object (บริเวรณสี่เหลี่ยมใช้แทนสิ่งของ) ได้
  - สามารถเซฟแผนที่เป็นไฟล์ XML ได้

การเซฟแผนที่เป็น XML

- เลือก Edit → Preferences..
- เปลี่ยน Store tile layer data as: เป็น XML